```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('/content/diabetes 2.csv')

print("Dataset Overview:")
print(df.head())
print("\nDataset Info:")
print(df.info())
print("\nDataset Description:")
print(df.describe())
```

```
Dataset Overview:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \
0            6      148             72             35        0  33.6

1            1       85             66             29        0  26.6

2            8      183             64              0        0  23.3

3            1       89             66             23       94  28.1

4            0      137             40             35      168  43.1


   Pedigree  Age  Outcome
0     0.627   50        1
1     0.351   31        0
2     0.672   32        1
3     0.167   21        0
4     2.288   33        1

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Pregnancies    768 non-null    int64
 1   Glucose        768 non-null    int64
 2   BloodPressure  768 non-null    int64
 3   SkinThickness  768 non-null    int64
 4   Insulin        768 non-null    int64
```

```
 5   BMI             768 non-null    float64
 6   Pedigree        768 non-null    float64
 7   Age             768 non-null    int64
 8   Outcome         768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None

Dataset Description:
       Pregnancies     Glucose  BloodPressure  SkinThickness
Insulin  \
count   768.000000  768.000000     768.000000     768.000000
768.000000
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI    Pedigree         Age     Outcome
count  768.000000  768.000000  768.000000  768.000000
mean    31.992578    0.471876   33.240885    0.348958
std      7.884160    0.331329   11.760232    0.476951
min      0.000000    0.078000   21.000000    0.000000
25%     27.300000    0.243750   24.000000    0.000000
50%     32.000000    0.372500   29.000000    0.000000
75%     36.600000    0.626250   41.000000    1.000000
max     67.100000    2.420000   81.000000    1.000000
```

```python
X = df.drop('Outcome', axis=1)
y = df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
print(f"\nTraining set size: {len(X_train)}")
print(f"Testing set size: {len(X_test)}")
```

```
Training set size: 614
Testing set size: 154
```

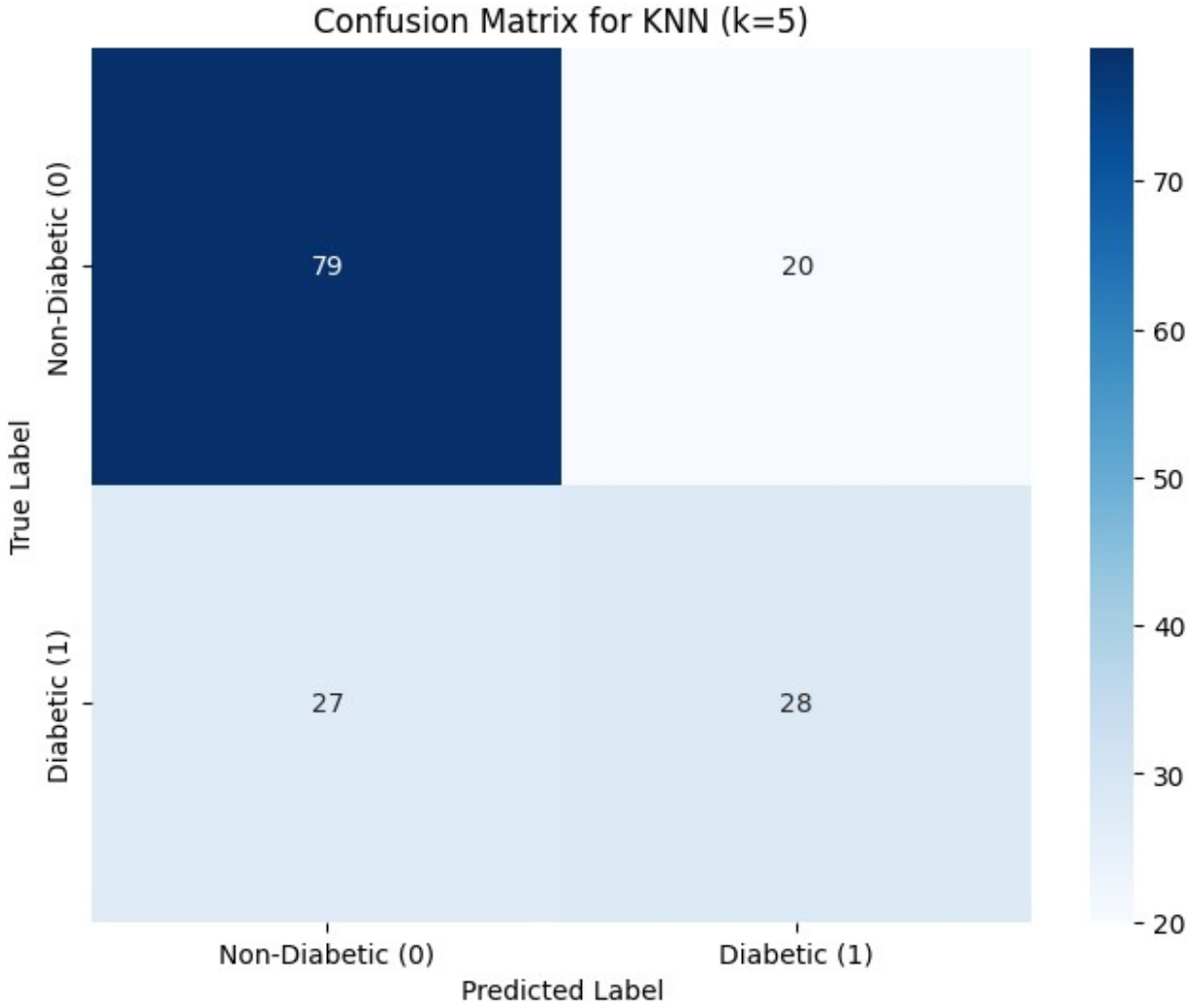
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

k = 5
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train_scaled, y_train)
```

```
KNeighborsClassifier()
```

```
y_pred = knn.predict(X_test_scaled)

cm = confusion_matrix(y_test, y_pred)
print("\n" + "="*50)
print("CONFUSION MATRIX")
print("="*50)
print(cm)
print("\nConfusion Matrix Breakdown:")
print(f"True Negatives (TN): {cm[0][0]}")
print(f"False Positives (FP): {cm[0][1]}")
print(f"False Negatives (FN): {cm[1][0]}")
print(f"True Positives (TP): {cm[1][1]}")


==================================================
CONFUSION MATRIX
==================================================
[[79 20]
 [27 28]]

Confusion Matrix Breakdown:
True Negatives (TN): 79
False Positives (FP): 20
False Negatives (FN): 27
True Positives (TP): 28

accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("\n" + "="*50)
print("EVALUATION METRICS")
print("="*50)
print(f"Accuracy: {accuracy:.4f} ({accuracy*100:.2f}%)")
print(f"Error Rate: {error_rate:.4f} ({error_rate*100:.2f}%)")
print(f"Precision: {precision:.4f} ({precision*100:.2f}%)")
print(f"Recall: {recall:.4f} ({recall*100:.2f}%)")


==================================================
```

```
EVALUATION METRICS
==================================================
Accuracy: 0.6948 (69.48%)
Error Rate: 0.3052 (30.52%)
Precision: 0.5833 (58.33%)
Recall: 0.5091 (50.91%)
```

```python
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Non-Diabetic (0)', 'Diabetic (1)'],
            yticklabels=['Non-Diabetic (0)', 'Diabetic (1)'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title(f'Confusion Matrix for KNN (k={k})')
plt.show()
```



```python
results_df = pd.DataFrame({
    'Metric': ['Accuracy', 'Error Rate', 'Precision', 'Recall'],
    'Value': [f"{accuracy:.4f}", f"{error_rate:.4f}",
f"{precision:.4f}", f"{recall:.4f}"],
    'Percentage': [f"{accuracy*100:.2f}%", f"{error_rate*100:.2f}%",
                   f"{precision*100:.2f}%", f"{recall*100:.2f}%"]
})

print("\n" + "="*50)
print("SUMMARY TABLE")
print("="*50)
print(results_df.to_string(index=False))
```

```
==================================================
SUMMARY TABLE
==================================================
    Metric  Value Percentage
  Accuracy 0.6948     69.48%
Error Rate 0.3052     30.52%
 Precision 0.5833     58.33%
    Recall 0.5091     50.91%
```